Victorian 6502 User Group Newsletter

KAOS

For People Who Have Got Smart

OSI SYM KIM AIM UK101 RABBLE 65

Registered by Australia Post
Publication No. VBG4212

Vol.5 No.8 May 1985

Changes to the club structure continue and the newsletter heading will be altered shortly. Meanwhile, address all correspondence to John Whitehead, . All material for the newsletter should be sent to Brian Busby,

Club funds are low, and a fee of 50 cents will be levied at each meeting. Also, for those who have it, a coffee contribution of 50 cents will cover costs.

We are glad to report that Ian Eyles is home again from hospital.

Ed Richardson still has parallel printer boards available for OSI machines. Merely send him $1 for pcb, component layout and instructions.

The next meeting will be at 2pm on Sunday 26th May at the Essendon Primary School on the corner of Raleigh and Nicholson Streets, Essendon. The school will be open at 1pm.

The closing date for articles for the June newsletter is the 7th June.

DISK UNBUCKLING.

Do you have problems with Mitsubishi 5"(8" lookalike) drives? First, they demand special disks which are expensive and apparently scarce. Second, those disks you can get usually have no hub ring reinforcement and are made of very thin plastic which buckles and flaps while spinning, unless you were lucky and the disk centered first time. Eventually the buckling disappears - after a week or two - so the disk is still usable. A quicker way to restore them is to gently warm the centre. The best way to do this is in the sun. Cut a 4 cm. hole in a piece of white paper and place it over the disc to protect all but the centre while exposing it to the sun. In summer, the disk flattens in a few seconds but takes longer in fall. Maybe an electric fire would do but great care should be taken to avoid excessive heating and probably further buckling! Needless to say, dust must be avoided all the time. (BJB)

INDEX

The Meeting Was KAOS

There was once again a pretty good roll-up for the April meeting. About a dozen hard-core hackers brought their machines, including at least 2 Rabbles,3 Macs, an OSI or two, an Atari and lots of Apple clones. There were also quite few new and also ancient faces in the crowd. Welcome to the new, and welcome back to the rest! In particular, we were honoured by the presence of Eric Lindsay.

Ray Gardner showed his 6511AQ Universal Controller board. Very impressive, Ray, but can it play football? The price of the board and instructions is most reasonable ($25 I believe) for what is really a very useful item.

The meeting was co-chaired by Ron Cork and Jeff Rae and the first subject to come up was money, or rather, the lack of it in our club coffers. A lot of ideas on how to raise money were kicked around, but as I understand it the probable result is that there may be a small door charge, say 50 cents, and the subscription fee will be going up as of the next year.

One serious subject that must be mentioned is the growing practice of 'backing up' copyright software in public. It is illegal and we are requesting those engaging in this practice to please desist before problems develop for the club. By all means demonstrate any new software, that is what the club is supposed to be all about,the exchange of information and ideas, especially new ideas. Those of you who want to get copies of particular items of software, by all means form special interest groups, exchange names and addresses and get together in private, what you do there is entirely up to you. Please don't think we are being spoilsports, we are just trying to prevent trouble. Any public domain software can of course be copied without restrictions.

Last but not least, I am adding my voice to the call for more of you, the members of KAOS, to get off your behinds and help in the running of KAOS for our (the club, that is) mutual benefit. You only get out of it what you are prepared to put into it and we can't do it properly without everybody pitching in. I shall leave you with that thought and see you at the meeting.

R.M.H.

FOR SALE

RABBLE 65, 1 NPI B51 DISK DRIVE, POWER SUPPLIES (12 & 5 VOLT), PRINTER OUTPUT, MONITOR (BMC), 1 BOX OF DISKS, COMPDOS 1.3 AND 1.2, SARGON CHESS, WP6502, SOME GAMES. $700.00 ONO.

OSI SUPER BOARD, TASAN VIDEO BOARD, TASKER BUSS, PRINTER OUTPUT, IN CASE (COMPSOFT), 5 VOLT POWER SUPPLY, SOME SOFT WARE.
$ 250.00 ONO
KEVIN CHARLESON
PHONE

THE ENHANCED APPLE IIe

APPLE has announced that a new enhanced version of the IIe is now available (in the U.S.A. at least).

The new APPLE IIe is achieved by the replacement of four integrated circuits. These changes bring a lot of the features of the IIc onto the IIe.

The changes are:-

* New microprocessor - the 65C02 with 27 additional instructions.

* A character generator ROM that allows the IIe to display Mouse TEXT characters.

* Two new Monitor ROMs that give the IIe better interrupt support and add a collection of advanced programming features.

Also included is a built-in mini-assembler for machine-language programming similar to that which the original Integer Basic of Apple II had.

In the U.S.A. these four ICs are made available to update existing IIe's, and are installed by the dealers for about $70 U.S.

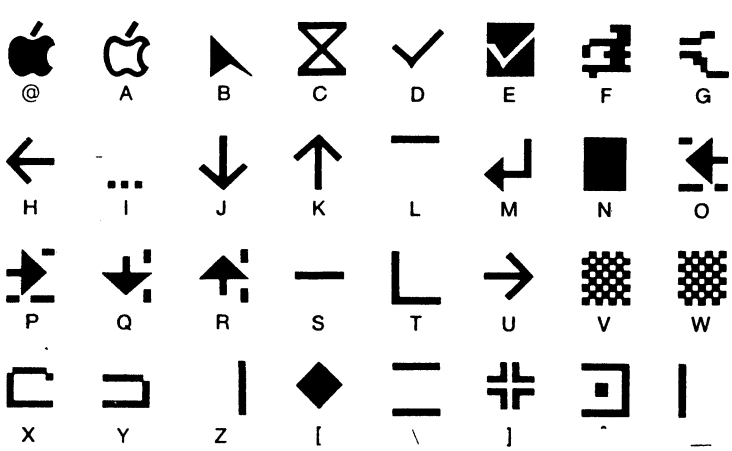
The enhanced APPLE IIe will enable you to utilise Mouse Text which is required for all Mouse driven software. The type of software expected to be developed will have Macintosh type features, such as windows and pull-down menus.

Apple states that this upgrade will mean losing a small amount of compatibility with current generation IIe software, but they advise IIe owners to upgrade for the next generation of software which is expected to be developed.

This upgrade of the IIe will make the IIe and IIc more compatible and hopefully encourage software developers to take up the challenge of providing new Mouse driven Mac-type software.

Simulated MouseText characters and their corresponding ASCII characters are shown in Figure 5-1.

MouseText Characters



APPLE SOFTWARE

JACKPOT BY R.WOODHOUSE

The program "JACKPOT" is a computer simulation of a poker machine.As such the odds are fairly consistent with the real thing.

All reels spin when you press the space bar and all winnings are credited to your score.

If you win, there is a click for every coin deposited to your score.

As with the real thing, you can't really win, but have fun anyway.

```
1  REM  ******************
2  REM  * JACKPOT        *
3  REM  * BY R.WOODHOUSE *
4  REM  *                 *
5  REM  * FOR APPLE ][    *
6  REM  *       48K       *
7  REM  *    10/4/1985    *
8  REM  ******************
10  HOME
20 D = 100
30 A$ = "9J*$JA9K*K$9AQJ*9Q*JQA9$K*K9AJ$*J9"
40 B$ = "$*AKJ9Q9*$KAQ9J9*$9J9QAK$*9Q9JKA*$"
50 C$ = "J9*KQ$AQJ9$*JK*J*9J*KJ*$9JQA$QK*9J"
60 A =  INT ( RND (1) * 34) + 1
65  IF A < 5 THEN 60
70  REM  TITLE
80  VTAB 10: HTAB 2: INVERSE : PRINT "                                    "
     : NORMAL
90  PRINT
100  HTAB 13: PRINT "**  JACKPOT  **"
110  PRINT
120  HTAB 2: INVERSE : PRINT "                                          ": NORMAL

130  FOR X = 1 TO 2000: NEXT X
135  HOME
140  VTAB 4: HTAB 4: PRINT "DO YOU WANT INSTRUCTIONS? (Y/N)
145  INPUT " > ";Z$: IF  LEFT$ (Z$,1) = "Y" THEN  GOSUB 10000
147  HOME
150  REM  SET UP SCREEN
160  VTAB 6: HTAB 5: INVERSE : PRINT "                    "
170  HTAB 5: PRINT "  ": NORMAL : PRINT "   ": INVERSE : PRINT "  ": NORMAL
     : PRINT "   ": INVERSE : PRINT "  ": NORMAL : PRINT "   ": INVERSE
     : PRINT "  "
180  HTAB 5: PRINT "  ": NORMAL : PRINT "   ": INVERSE : PRINT "  ": NORMAL
     : PRINT "   ": INVERSE : PRINT "  ": NORMAL : PRINT "   ": INVERSE
     : PRINT "  "
190  HTAB 5: PRINT " -": NORMAL : PRINT "   ": INVERSE : PRINT "--": NORMAL
     : PRINT "   ": INVERSE : PRINT "--": NORMAL : PRINT "   ": INVERSE
     : PRINT "- "
200  HTAB 5: PRINT "  ": NORMAL : PRINT "   ": INVERSE : PRINT "  ": NORMAL
     : PRINT "   ": INVERSE : PRINT "  ": NORMAL : PRINT "   ": INVERSE
     : PRINT "  "
210  HTAB 5: PRINT "  ": NORMAL : PRINT "   ": INVERSE : PRINT "  ": NORMAL
     : PRINT "   ": INVERSE : PRINT "  ": NORMAL : PRINT "   ": INVERSE
     : PRINT "  "
220  HTAB 5: PRINT "                    ": NORMAL
390  VTAB 5: HTAB 28: PRINT "$ $ $ = 200"
400  HTAB 28: PRINT "A A A = 100"
410  HTAB 28: PRINT "K K K = 100"
420  HTAB 28: PRINT "Q Q Q = 100"
430  HTAB 28: PRINT "J J J =  50"
440  HTAB 28: PRINT "* * * =  50"
450  HTAB 28: PRINT "9 9 9 =  50"
460  HTAB 28: PRINT "$ $ . =  50"
470  HTAB 28: PRINT "A A . =  30"
480  HTAB 28: PRINT "K K . =  30"
490  HTAB 28: PRINT "Q Q . =  30"
500  HTAB 28: PRINT "J J . =  20"
```

```
510  HTAB 28: PRINT "* * . =  20"
520  HTAB 28: PRINT "9 9 . =  20"
530  HTAB 28: PRINT "$ . . =  10"
1490  VTAB 2: HTAB 5: PRINT "YOU HAVE ";D;" COINS LEFT "
1500  VTAB 20: HTAB 2: PRINT "PRESS ANY KEY TO START"
1510  GET Z$
1520 D = D - 10
1530  VTAB 20: HTAB 2: PRINT "                              "
1540  VTAB 2: HTAB 5: PRINT "YOU HAVE ";D;" COINS LEFT "
1550  IF D < 0 THEN 2500
1580 A =  INT ( RND (1) * 16) + 1
1590  IF A < 5 THEN 1580
1600  FOR N = 1 TO A
1605 O = N +  INT ( RND (1) * 5)
1607 P = N +  INT ( RND (1) * 5)
1610 D$ =  MID$ (A$,N,1)
1612 E$ =  MID$ (A$,N + 1,1)
1615 F$ =  MID$ (A$,N + 2,1)
1620  VTAB 7: HTAB 8: PRINT D$
1622  VTAB 9: HTAB 8: PRINT E$
1625  VTAB 11: HTAB 8: PRINT F$
1710 G$ =  MID$ (B$,O,1)
1720 H$ =  MID$ (B$,O + 1,1)
1730 I$ =  MID$ (B$,O + 2,1)
1740  VTAB 7: HTAB 13: PRINT G$
1750  VTAB 9: HTAB 13: PRINT H$
1760  VTAB 11: HTAB 13: PRINT I$
1800 J$ =  MID$ (C$,P,1)
1810 K$ =  MID$ (C$,P + 1,1)
1820 L$ =  MID$ (C$,P + 2,1)
1830  VTAB 7: HTAB 18: PRINT J$
1840  VTAB 9: HTAB 18: PRINT K$
1850  VTAB 11: HTAB 18: PRINT L$
1860  NEXT N
1900 W = 0
1905 W$ = E$
1910  IF W$ = "$" THEN W = 10
1915 W$ = W$ + H$
1920  IF W$ = "$$" THEN W = 50
1925  IF W$ = "AA" THEN W = 30
1930  IF W$ = "KK" THEN W = 30
1935  IF W$ = "QQ" THEN W = 30
1940  IF W$ = "JJ" THEN W = 20
1945  IF W$ = "**" THEN W = 20
1950  IF W$ = "99" THEN W = 20
1955 W$ = W$ + K$
1960  IF W$ = "$$$" THEN W = 200
1965  IF W$ = "AAA" THEN W = 100
1970  IF W$ = "KKK" THEN W = 100
1975  IF W$ = "QQQ" THEN W = 100
1980  IF W$ = "JJJ" THEN W = 50
1985  IF W$ = "***" THEN W = 50
1990  IF W$ = "999" THEN W = 50
2380  IF W > 1 THEN  VTAB 15: HTAB 5: PRINT "YOU WIN ";W;" COINS": FOR N =
    1 TO 1000: NEXT N
2390 D = D + W
2400  GOSUB 5000
2450  VTAB 15: HTAB 5: PRINT "                    "
2480  IF D < 0 THEN 2500
2490  GOTO 1490
2500  HOME : VTAB 4: HTAB 4: PRINT "SORRY,YOU'RE BROKE.NO CREDIT"
2510  END
5000 S =  - 16336
5010  FOR X = 0 TO W
5020 SOUND =  PEEK (S) -  PEEK (S) +  PEEK (S) -  PEEK (S) +  PEEK (S) -  PEEK
    (S)
5030  FOR I = 1 TO 50: NEXT I
5040  NEXT X
5050  RETURN
10000  VTAB 4: HTAB 3: PRINT "YOU HAVE 100 COINS TO START WITH AND    EACH
    TURN COSTS YOU 10 COINS
10010  HTAB 3: PRINT "ANY WINNINGS ARE LISTED ON THE SCREEN   AND ARE AUTO
    MATICALLY CREDITED TO YOUR  SCORE"
10020  PRINT
10030  HTAB 15: PRINT "GOOD LUCK!!!!!"
10040  FOR X = 1 TO 5000: NEXT X
10050  RETURN
```

SUPERBOARD

©May 1985.

Newsletter of the Ohio Superboard User Group, 146 York St, Nundah, 4012.

MORE ON RANDOM ACCESS FILES

In KAOS 5/1, I developed some useful routines to construct an efficient random access files program.
In this article, I build on these routines to demonstrate a practical program, as user friendly as possible. What exactly do I mean by user friendly?

(1) A minimum of computer knowledge should be needed to operate it.
(2) The program must be as error proof and crash proof as possible.
(3) A minimum of keystrokes should be needed to update the records.

The program as presented keeps jockey records. I keep lots of figures on jockey's performances, and need five basic numbers to work them all out. I keep a year's figures on each jockey, updated each month. When I enter the figures for June 85, I subtract the June 84 figures. Some thirteen files are needed for this, and I don't intend to bore you with repetition, so I'll just stick to the basic program.
When all added, and reduced to strings(removes those unneccessary spaces in front of numbers), each record requires 27 bytes.

Only changing information is kept on data files, and my current tally is 90 riders, which will fit easily on one 8" disk track. Jockey's names and associated record numbers are in Basic data statements. The names are in alphabetical order for each state, so that sorting is never required.

Only changes to the KAOS 5/1 program are explained here. All the basic techniques are provided so that setting up a program for your purpose should be easy.
In line 20, the pokes to 2888 and 8722 allow a carriage return input. This is used to improve the speed of data entry. A jockey might have ten rides for every win, so it is often only necessary to update some of the fields. In this program, a carriage return leaves the old figure intact.
Line 30 has a couple of new variables. PR is for printer. A parallel printer OSI style is device 4, while most of you will use device 1. SC is for screen output. For most, this will be 2. A serial terminal user will have 1.
Lines 90-95 try to match your input to one of the names in the data statements, to locate the record number. This routine could be better written to give a faster data search.
Line 205 has a special check to prevent writing of a record outside the record number range. In any program using data files, data integrity is of paramount importance. Reading an incorrect track wouldn't matter much - writing it certainly would! The NULL0 in line 200 is also very important. Writing nulls to disk records will overwrite other records, leaving you with a terrible mess.
The test write routine, lines 500-520, has been improved so that only tracks in the record range are written to. The KAOS 5/1 routine went beyond these tracks.
The printer routine and mathematical calculations, lines 600-699 will need to be modified to produce whatever data you want. There is an option for printing to the screen only.

65D3.2 programs can only have so much protection. The program prevents exiting by normal means, but unlike 65U, there is no on error goto function. Doing something like removing a disk, then ending the program, will cause an exit to command mode. Should this happen, and you have a buffer full of new information, then GOTO999 will ensure storage.
The write test records lines, 500 to 520, should be deleted once a working program is established.

SUPERBOARD

```
5 REM POKE120,1:POKE121,64:POKE16384,0:NEW before typing in program!
10 REM RIDERS RECORDS ON RANDOM ACCESS FILES
20 POKE2073,96:POKE2888,0:POKE8722,0:REM No exits allowed
25 M=51:P=256:OL=9105:OH=9106:IL=9098:IH=9099
30 RL=27:NR=200:FT=6:NP=13:DD=1:DD$="A":PR=1:SC=2:DV=SC:REM See text
35 RT=INT(NP*P/RL):CR$=CHR$(13):NP$=STR$(NP):IFNP=13THENNP$="D"
40 PRINT"Insert data disk#"DD"in drive "DD$:INPUT"Ready";A$
45 DISK!"SE "+DD$
49 REM * Main menu *
50 PRINTCHR$(26);"RIDER'S RATING PROGRAM":PRINT:PRINT
55 RESTORE:PRINT:PRINTTAB(3)"OPTIONS TO ENTER":PRINT
60 PRINT"Rider's name for search":PRINT
65 PRINT"Printout rider's ratings (P)":PRINT
70 PRINT"Finish with program (F)":PRINT:PRINT
75 INPUT"Your choice";A$:A=LEN(A$):IFA$=""GOTO50
80 IFA$="P"THENGOSUB600:GOTO50
85 IFA$="F"GOTO 999
90 READN$,RN:IFN$="END"THENPRINT"Can't find ";A$:GOTO55
95 IFA$<>LEFT$(N$,A)GOTO90
99 REM * Input new data *
100 GOSUB300:PRINTCHR$(26):PRINT:PRINT"Press <RETURN> for no change"
110 GOSUB400:PRINT:PRINT"Record for rider ";N$:PRINT:PRINT
120 PRINT"Rides"TAB(15)S$;:INPUTA:GOSUB180:IFA<>OTHENS$=A$
130 PRINT"Wins"TAB(15)W$;:INPUTA:GOSUB180:IFA<>OTHENW$=A$
140 PRINT"Places"TAB(15)P$;:INPUTA:GOSUB180:IFA<>OTHENP$=A$
145 PRINT"BTC Return"TAB(15)R$;:INPUTA:GOSUB180:IFA<>OTHENR$=A$
150 PRINT"Level Return"TAB(15)L$;:INPUTA:GOSUB180:IFA<>OTHENL$=A$
155 A$=S$+W$+P$+R$+L$:IFLEN(A$)+5>RLTHENPRINT:PRINT"ERROR!":GOTO110
160 GOSUB200:GOTO50
179 REM * Convert data to string *
180 A$=STR$(A):IFASC(A$)=32THENA$=RIGHT$(A$,LEN(A$)-1)
185 PRINT:RETURN
199 REM * Write record to buffer *
200 POKEOL,L:POKEOH,M+H:NULL0
205 IFRN>NRTHENPRINT"Only"NR" records available":GOTO55
210 PRINT#5,S$;CR$;W$;CR$;P$;CR$;R$;CR$;L$;CR$;:PRINT#9:WF=1:RETURN
299 REM * Record Read/Write/Find *
300 NT=FT+INT(RN/RT):IFNT=PT GOTO330
310 IFWF=1THENDISK!"SA "+RIGHT$(STR$(100+PT),2)+",1=3300/"+NP$:WF=0
320 PT=NT:DISK!"CA 3300="+RIGHT$(STR$(100+PT),2)+",1"
330 B=(RN-(PT-FT)*RT)*RL:H=INT(B/P):L=B-H*P:RETURN
399 REM * Input record *
400 POKEIL,L:POKEIH,M+H
410 INPUT#5,S$,W$,P$,R$,L$:RETURN
499 REM * Write test records, enter line 47 GOTO 500
500 FORR=M*PTOM*P+NP*P:POKER,0:NEXT:FORNT=FTTOFT+INT(NR/RT)
505 DISK!"SA "+RIGHT$(STR$(100+NT),2)+",1=3300/"+NP$:NEXT
510 S$="0":W$=S$:P$=S$:R$=S$:L$=S$:FORRN=OTONR
520 GOSUB300:GOSUB200:NEXT:PRINT"REMOVE LINE 47":GOTO999
599 REM * Print routine *
600 INPUT"Date";N$:INPUT"Want printer";A$:IFASC(A$)=89THENDV=PR
605 PRINT#DV,TAB(15)"RIDER'S RECORDS 1/8/84 TO ";N$
610 READN$,RN:IFN$="END"THENPRINT:DV=SC:RETURN
615 IFASC(N$)=42THENA$=N$:PRINT#DV:PRINT#DV,A$:READN$,RN
620 GOSUB300:GOSUB400:S=VAL(S$):IFS=0GOTO610
625 W=VAL(W$):PL=VAL(P$):R=VAL(R$):L=VAL(L$):PRINT#DV,N$;TAB(20)S;
630 PRINT#DV,TAB(25)INT(W*100/S+.5);TAB(30)INT(PL*100/S+.5);
635 PRINT#DV,TAB(35)INT(L*100/S);TAB(40)INT(R/(W*100-R)*100);
```

THE CPM FILE

Here we are again for May. The year is just zipping past and all the things I was going to get done seem to be as far away as ever - you probably know the feeling too! I am a little concerned that I am getting no feedback from those of you out there lucky enough to have the CPM expansion for OSI/RABBLE. Is nobody doing anything interesting with their machines? If you are, how about telling the rest of us all about it, we need articles for KAOS.

For that matter, how are you CPM hackers getting on with the new system after all these months? How are you finding the software situation? Are you happy, disappointed, what? Does anybody out there need any help, what problems have you got? If any of the above apply, tell us, we can't do anything if we don't know and we can't function effectively as mushrooms. After all, this is your file, I only write it.

At the risk of being called a traitor, I have a suggestion for those of you who don't have CPM and would like to but can't get the cards. The price of a fully assembled Pulsar LBB has been dropped to $450.+tax. The BIOS sources are available to permit customizing for drives, terminal width, etc. So if you get a LBB and use your OSI or RABBLE as a terminal, hey presto - CPM. Not as good as the real thing but better than nothing. Unless of course, somebody comes up with a fresh batch of CompSoft cards. Ray, how about an 8.4 monitor for the RABBLE with an extended terminal routine (intelligent) to fill the holes, or maybe a new ROM3 which could be a terminal/spooler arrangement (with built-in modem routines) for use with boards like the LBB.

```
640 PRINT#DV,TAB(45)INT(1000/((W*100-R)/S))/10-1;:IFW=0GOTO650
645 PRINT#DV,TAB(52)INT(10*(S+L)/W)/10-1;
650 PRINT#DV:GOTO610
999 GOSUB310:DISK!"SE A":POKE2073,173:POKE2888,27:POKE8722,27:END
1000 DATA*BRISBANE RIDERS   RIDES  W%   P%  LSP% TPP% AV SP  W SP,0
1003 DATAG.BIRRER,2
```

If you have any suggestions to improve the program, or queries about it, I'd be pleased to hear from you.

Meeting of the KAOS (Qld) User Group - 28th April 1985.

Attendance: 10 Computers: 3

The meeting was late to start, so late in fact that I thought it finally had happened, and nobody was coming. I am delighted to advise that OSI had made a strong comeback - all three machines were OSI, not a BBC in sight!

Bob Best had some new music tunes for his C4P. Paul Gray and Doug Robinson, both working on their disk systems, had come up against a snag. Most of the disks that Bob had loaned them wouldn't display on their C1's. Nobody seemed quite sure why not, or what to do about it. Some disassembly of the DOS was done without much success. It was generally agreed that the different screen and keyboard scan were at fault - but what to do about it?

Bernie Wills had his machine in, plus a pile of electronic bits for sale. Some 2732's and a sound chip were snapped up at bargain prices. Bernie also offered a pile of magazines to anyone interested. Ian Mackenzie, Paul, and Doug shared them out.
Ian said nothing had been seen of the Sperry computers that had won the Education contract. Apparantly Apple had expected to get the nod. Heads will roll.

Some fun and games were had getting a Qume daisywheel interfaced to a C1P. More so with a Frogger lookalike game which had an unknown bug causing premature frog genocide. Ross Beneke promised to solve it. The meeting ended at 4.30pm.

SERIAL BACKSPACE FIX

Some months back, a member asked for some help to get his terminal to backspace properly with OS65D. Apparently the terminal used Hex 15 instead of the standard Hex 8.
Try adding these two pokes to your BEXEC* and then put it back to disk.

```
POKE 1424,21:POKE1434,21:REM Backspace serial terminal
```

GETTING MORE NULLS

Most ROM Basic users would know that you can poke location 13 decimal to get any number of nulls up to 255. OSI Basic only allows NULL8, and often this is not enough.

For OS65D users, the location is 21 decimal. You can simply poke it to get all the nulls you want, or permanently change the compare function by poking location 2164 decimal with a bigger number, say 50, from a line in BEXEC*. After that, a NULL20 will work just fine!

HELP WANTED

OSUG have had some queries re additional facilities offered by OS65D4.1 over the 3.2 or 3.3 DOS. I would greatly appreciate an article on this subject from any member who has used 4.1.

Ed Richardson.

FORTH PART TEN EXTENDING THE FORTH COMPILER

by Ray.Gardiner.

Many languages allow the programmer to extend the language by creating procedures that can be invoked by name or called from a subroutine library. Forth allows you to progress up the ladder a step above all the others. You can actually define words that define words. How to write a compiler in 25 words or less.

Over the last few months we have mentioned the <BUILDS ... DOES> construction when discussing defining words. This construct is possibly the most powerfull available in any language. Consider the following example which might appear in the definition of a music language. We wish to refer to notes by their symbolic names, e.g. C# will represent C-sharp and so on. The note durations will be represented by, W,H,Q,E, etc. for whole, half, quarter, etc. The voice will be selected by V1...Vn for N-voices. Consider the definition of a word called NOTE as follows.

```
: NOTE  <BUILDS , DOES> @ PLAY-NOTE ;
```

We can use this to create new note definitions as follows,

```
 885 NOTE C#
 764 NOTE F#
```

when you type C#, the appropriate note will be played.
now you can go on to define chords and complete musical phrases that can be combined to make an entire composition.

```
: CHORD V1 C# V2 F V3 G# ;    ( No, I have no idea what that sounds like!)
```

At the most elementary level, the <builds...does> construction will save an enormous amount of typing by generating the defining template as the example above illustrates.

The construction can be divided into two seperate sections, firstly, the portion between the <builds and the does> is executed at compile time when you are compiling the new word. And the section of code between does> and the ; is executed when the newly defined word is later executed.

Let's call our defining word PARENT and the words created by PARENT will be called children. (why not?)

```
 : PARENT   <BUILDS  ( template ) DOES>  ( action )  ;

  PARENT  CHILD       ( executes template defines CHILD ),

  CHILD               ( executes action, ie. code between DOES> and ; )
```

Since the template and the action can be any forth structures the structure effectively allows us to create a whole new class of data structures with built in algorithms.

Such is the power and simplicity of the <builds does> construction that it is unlike anything available in conventional languages. Recently a new breed of object oriented languages has attracted some publicity, and considerable portions of the smalltalk philosophy has been engineered into the MACINTOSH user interface. The creation of intelligent data structures using the <builds does> construction is relatively easy.

Consider as an example a self averaging array that collects its own data.

```
: SMART <BUILDS
                INITIALIZE.ARRAY  ALLOT.MEMORY
         DOES>
                COLLECT.DATA  CALCULATE.AVERAGE ;

SMART PAYROLL   ( defines payroll, initializes array and reserves memory )

PAYROLL  CHEQUES ( collects data, calculates averages, prints cheques etc.)
```

At execution time DOES> returns the address pointing to the start of the array.

Or what about a video game object that moves itself around the screen changing shape and avoiding obstacles.

```
: PLAYER  <BUILDS
                SHAPE.DATA.1 SHAPE.DATA.2 PLAYER.ATTRIBUTES
           DOES>
               ?FLAG  IF   SHAPE.1 CLEAR.FLAG
                      ELSE SHAPE.2 SET.FLAG
                      THEN PLAYER.MOVE ;

PLAYER  GHOST.MUNCHER
PLAYER  GOBLIN
PLAYER  SPACESHIP

: GAME  BEGIN
          UPDATE.SCORES
          GHOST.MUNCHER
          GOBLIN
          SPACE.SHIP
          HANDLE.USER.EVENTS
          ?GAME.OVER
        UNTIL ;
```

Each player created using the defining word PLAYER will have the same basic behaviour and attributes. So for each class of object the template is the same. (Each defining words creates a new class of OBJECTS)

The following is from the Rochester forth conference papers. It is possibly the most elegant piece of code ever written,

```
: ACTUAL <BUILDS , DOES> ;
: STRUCTURE <BUILDS DOES> ACTUAL ;
```

Structure is a defining word which when invoked defines other defining words using ACTUAL.....you now have the basis of a relational data base!!!!.

That concludes the 10 part series overview of the FORTH language, many thanks to those who have expressed interest and support during the series of articles. For the next few months it is hoped to present applications and various utilities.

OSI Keyboard
by John Whitehead

The OSI keyboard is in the form of a matrix. It is accessed by sending data between Ø and 255 to its memory location of 57Ø88 (hex $FDØØ) and then reading that location to detect which key or keys are pressed. Due to incomplete address decoding, it actually covers memory 57Ø88 to 57343.

For normal text use, a machine code routine in the monitor ROM at $FDØØ takes care of keyboard scanning and decoding. This routine leaves the ASCII value of the key pressed in the 65Ø2 accumulator. This routine can be called from basic with:-

```
POKE 11,Ø : POKE 12,253 : REM 253=$FD
X=USR(X) : A = PEEK (531) : PRINT CHR$(A)
```

The hardware for the superboard C1P is different to the C4P. The C4P drives one row high at a time where the C1P drives one row low at a time. This makes the peek and poke values different. Both are show in the chart but examples are for C1P.

For special use, such as game movement keys or joysticks, a simple BASIC or m/code routine can detect keypresses.

To detect a single key, e.g. the space bar, look up the row and column values for 'SPACE' in the chart and use them in the program below. The CTRL C routine has to be turned off as it will poke rows zero and two which could give wrong column values. If you only want to detect keys in row zero, you can leave CTRL C turned on and just peek the columns. The CTRL C routine is at $FF9B. You could disassemble this to see how it works.

```
1Ø KEY=57Ø88
2Ø POKE53Ø,1 :REM Turn off CTRL C.
3Ø POKE KEY,253 :REM Drive row two low
4Ø PRESS = PEEK(KEY) : PRINT PRESS: REM get column value
5Ø IF PRESS = 239 THEN PRINT"SPACE BAR" : POKE53Ø,Ø : STOP
6Ø REM 239 = Column four driven low by connecting it to row two
7Ø GOTO3Ø
```

To detect two or more keys pressed together, a logical and is performed on the row and column values from the chart.e.g. for CTRL Z, drive rows Ø and 1 low and look at columns 5 and 6. To calculate a logical and, the values need to be converted to binary. The logical and of Ø and Ø = Ø, Ø and 1 = Ø, 1 and 1 = 1.

```
The poke value is:-                The peek value is:-
CTRL = 254 = $FE = %1111 111Ø      CTRL = 191 = $BF = %1Ø11 1111
   Z = 251 = $FB = %1111 1Ø11         Z = 223 = $DF = %11Ø1 1111
                   ---------                          ---------
                   1111 1Ø1Ø = $FA = 25Ø              1ØØ1 1111
                                                        = $9F = 159
```

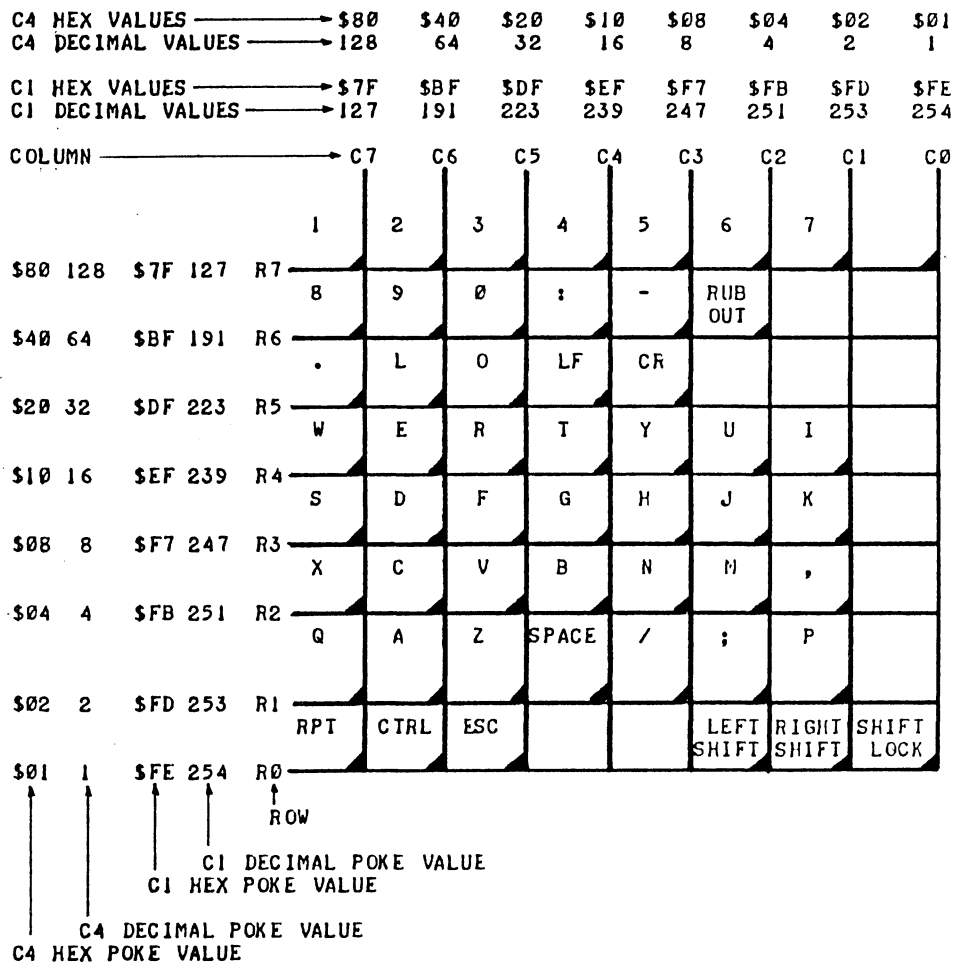
BASIC will do the above calculation for you as shown below.

```
1ØØ KEY=57Ø88
12Ø POKE53Ø,1
13Ø POKEKEY,254AND253
14Ø PRESS=PEEK(KEY):PRINTPRESS
15Ø IF PRESS=(191AND223) THEN PRINT"CTRL Z":POKE53Ø,Ø:STOP
17Ø GOTO13Ø
```

If you run the above, you may find you can't escape. this is because you have three keys pressed, one in row one and two in row zero and the column value is not 159. This can be fixed by changing line 15Ø to :-

```
IF PRESS = (191AND223) OR PRESS = (191AND223AND254) THEN....
```

PEEK(57088) OR $DF00 TO GET KEYPRESS VALUES



FOR SALE

Superboard Series II in metal case. 8K. Switchable Cegmon/Dabug; 1/2 MHz; 3/6/1200 baud. OSUG Basic 3 & 4. RS-232. B & W monitor. $100 ono.

Rabble board 34K RAM, fully populated except sound; some Eproms; disc interface operational; 40-way cable ; disc cable. $150 ono

Video board using 6545 controller; Cegmon (2716) and Basic 3 & 4 (2732) containing initialisation routine for 64 characters with C4 screen map. Can program 6545 for 24,32 or 80 chars. 40-way cable. $55 ono

Eprom burner with 28 pin Textool ZIF socket, plugs via 2 short cables into Rabble I/O ports. Needs approx. 30V supply but has regulator for 21 and 25V. Machine code software in eprom downloaded from socket to RAM at $C800 - suits Cegmon or use Basic program on cassette (both supplied). $40 ono

B. Wills,

ADDING ASM65 to SBII

by Gerard Campbell

For about a year now I have been using David Dodd's (DD) excellent modified and ROMable OSI assembler called 'ASM 65' ,which has many enhancements including more flexible renumbering ,unique auto-numbering ,extra commands such as FIND and COPY ,better editing (using DABUG) ,checksum loader and tape maker .ASM65 is available in a number of configurations at a reasonable price from DD. ASM 65 which allows full use of program RAM ,took a lot of development time and naturally DD is a bit disappointed due to the lack of response - only a few people are using it. Pherhaps this article might generate some interest amongst those people (youngsters ,newcomers ,students) with C1P ,DABUG cassette based systems. With ASM65 on board it is as easy to use as BASIC - encouraging use.

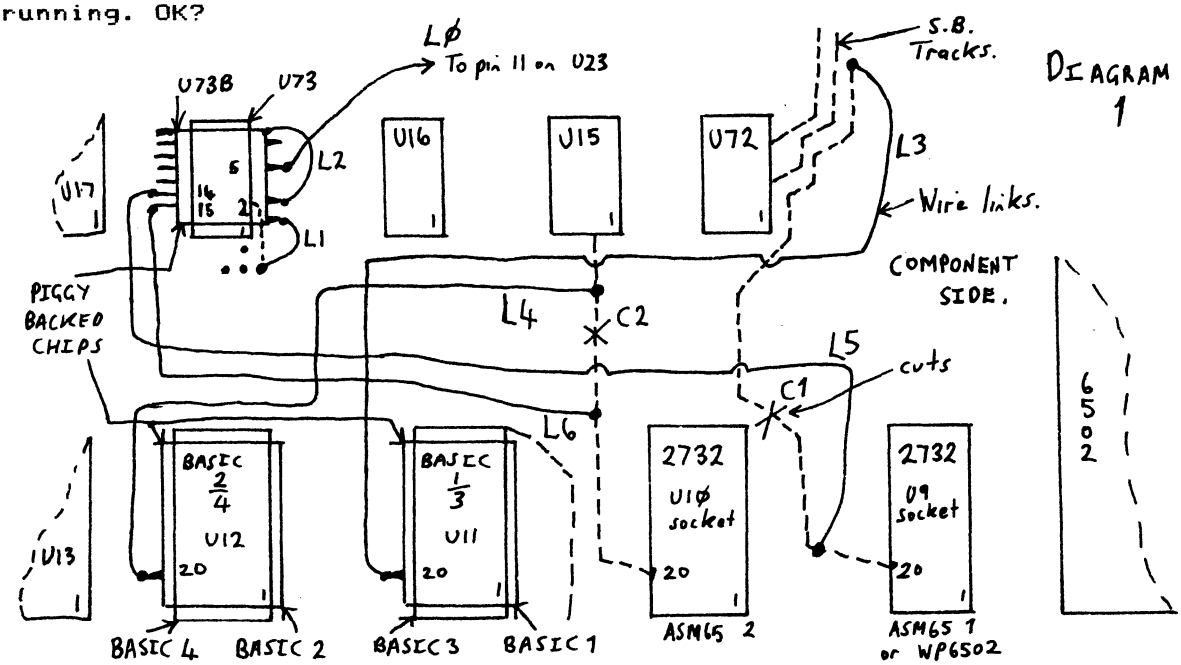
My version of ASM65 is at $8000-$9FFF and uses as program RAM $0300-$1FFF - plenty of memory for a fairly large program. With the addition of a VIA ,all the projects described in Rodnay Zaks "6502 Applications" are easily done and the ol' Super' becomes an ideal learning system. (It always was).

If any people reading this feel that the following mods are beyond them ,I would be willing to help (in Melbourne area) as I have some time to spare.

To add ASM65 to your C1P refer to Diagram 1. and SBII ccts and do the following --

1). Remove all Basic ROM's carefully.
2). Bend up pin 20 on BASIC 1 and piggyback on top of BASIC 3 (see HINT 1).
3).Bend up pin 20 on BASIC 2 and piggyback on BASIC 4.
4). Cut trace going from pin 20 on U9 socket towards U72 near pin 12 of U10. (C1).
5). Cut trace going from pin 20 on U10 towards U15 halfway b/w the two.(C2).
6). Put BASIC 2on4 in U12 socket ,BASIC 1on3 goes in U11 socket.
7). Solder lead ,L3, b/w pin 20 of BASIC 1 and U72 side of C1 (use plated through hole).
8). L4 goes b/w pin 20 of BASIC 2 and U15 side of C2.

At this point it might be wise to power up and see if BASIC is running. OK?



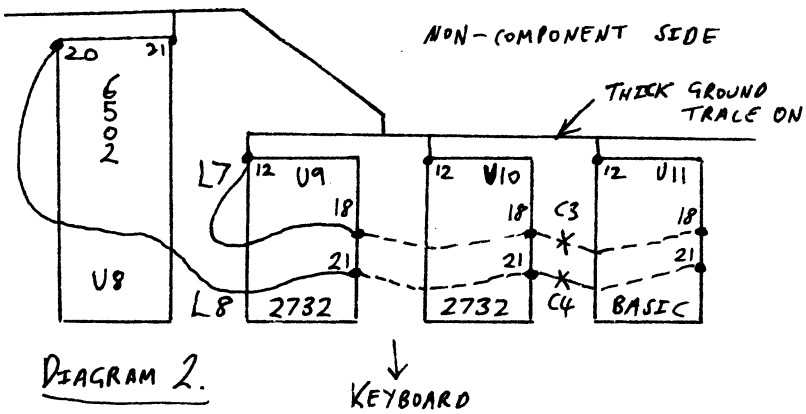
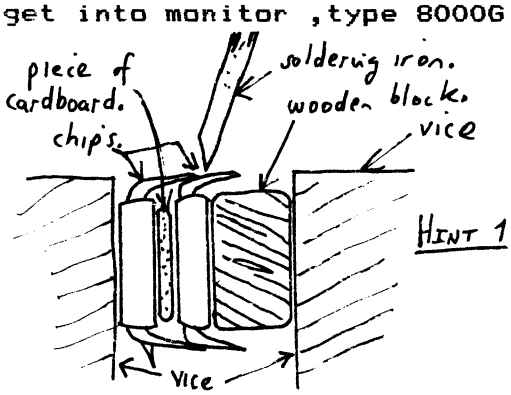


DIAGRAM 2.

9). Take a 74LS138 which we will call U73B, and bend up all pins except 3,4,6,8 and 16 ,and piggyback this on U73. Now solder the following leads.
10).L1 b/w pin 1 of U73B and pin 2 of U73 (use plated through hole).
11).L2 b/w pin 2 and pin 8 of U73B (i.e. ground B select line).
12).L5 b/w pin 14 of U73B and U9 side of C1.
13).L6 b/w pin 15 of U73B and U10 side of C2.
14).L0 b/w pin 5 of U73B and pin 11 of U23.
15).Put ASM65 1 in U9 and ASM65 2 in U10.

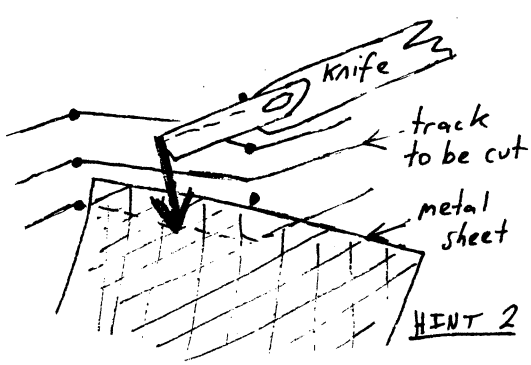
Now turn Superboard over and refering to Diagram 2 --
16).Cut traces joining pins 18 of ROMs b/w newly installed EPROMs and BASIC chips i.e. b/w U10 and U11 .This is cut C3. Do same for trace joining pins 21 ,this is cut C4. (see HINT 2).
17). Lead L7 goes b/w pin 18 and pin 12 of U9 i.e. ground chip enable thus using output enable as chip select.
18).L8 goes b/w pin 20 on U8 (tne 6502) and pin 21 of U9 ,i.e. join A11 to respective 2732 pin.

Now check solder joints ,piggy backing and trace cuts. Power up ,get into monitor ,type 8000G and happy assembling.



HINT 1

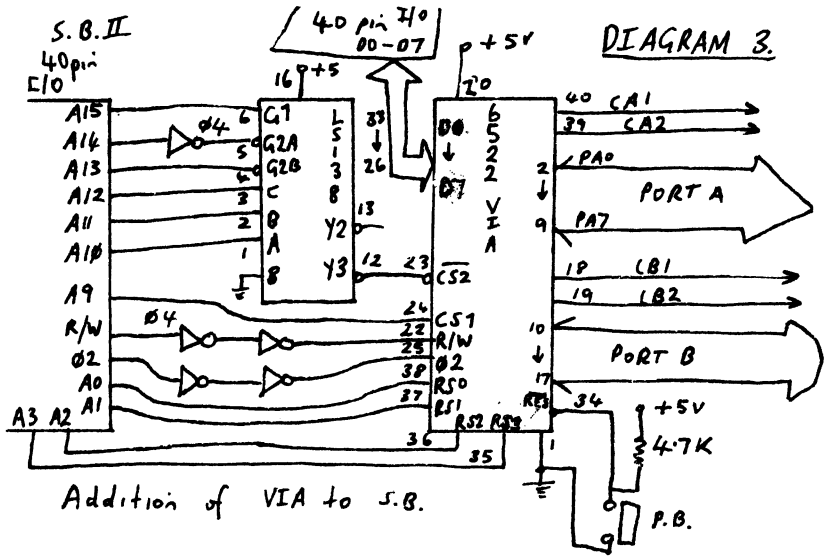
HINT 1. -- Hold chips lightly in vice or b/w two heavy objects. Pull cardboard out after soldering leaving an air gap which aids cooling.
HINT 2. -- Hold piece of thin metal covering neighbouring traces ,cut towards metal which stops accidental trace cutting.



HINT 2

*** Note that John Whiteheads WP6502 in ROM can be added to U9 socket instead of ASM65 1 ($8000-$8FFF).

To complete this article I will suggest an easy way to add a VIA to an unexpanded C1P. The circuit appearing in Diagram 3 uses 3 chips ,a 74LS04 ,a 74LS138 and the VIA.To use this cct you must do a naughty and bypass the 8T28 buffers U6 and U7 on the SBII. Do this by linking pins 2 to 3, 13 to 12 ,6 to 7 and 10 to 9 on each socket. This makes DD redundant (not David Dodd).
Make the cct on a piece of veroboard using a 40 pin wire wrap socket to plug it into SBII I/O. Bring the PIA ports out to molex pins ,16 pin sockets or whatever.



The cct decodes $CE00-$CFFF ,any address in this range ending with 0-F accesses the respective VIA resister. Another PIA or VIA can be added by connecting Y2 (pin 13) of the LS138 to its CS2 pin.(23). Its address will be within $CA00-$CBFF.
Phone nos.
DD's (BH) 03 6067729
Mine (AH) 03 4659325.

Superboard C1P monitor locations.
by John Whitehead and Mark Howell.

Original monitor or Dabug off	Dabug3	Dabug3J	Resmon	Cegmon
	!	!	!	!
Input vector at Ø218,9 FFBA	F8ØØ	F8ØØ	F8ØØ	FB46
Output vector at Ø21A,B FF69	FBØØ	FBØØ	FBØØ	FF9B
CTRL C vector at Ø21C,D FF9B	FF9B	FF9B	FF9B	FB94
Get keyboard value		FD21		
Printer flag			Ø217	
Screen clear	FCD5	FDC5	FCD5	FE59
Screen clear CHR$	127	127	127	26

The following are the same in all monitors above :-

NMI vector at FFFA,B	Ø13Ø	IRQ vector at FFFE,F	Ø1CØ
LOAD flag at Ø2Ø3	1=on	SAVE flag at Ø2Ø5	1=on
CTRL C flag at Ø212	Ø=on	Disc boot	FCØØ
ACIA reset	FCA6	Send char. to ACIA	FCB1
Reset vector at FFFC,D	FFØØ	Keyboard routine	FDØØ or FEED
M/Code monitor	FEØØ	Get char. from ACIA	FE8Ø
LOAD vector at Ø21E,F	FF8B	SAVE vector at Ø22Ø,1	FF96

Dabug uses memory at ØØEØ to ØØE8 and ØØF6 to ØØFF.
Cegmon uses memory at Ø222 to Ø234 that Dabug does not.

Major improvements over the original SYN6ØØ monitor are :-
Dabug 3. Single key BASIC, Screen editing, Screen clear, Screen freeze and screen swop for 48 x 12.

Dabug 3J. As Dabug 3 plus a correctly decoded keyboard. See KAOS Oct. 83.

Resmon. As Dabug 3 plus improved keyboard, Tape view, Printer flag. (No single key BASIC). See KAOS March 84.

Cegmon. Split screen, Screen editor and Screen clear, Improved keyboard, Hex save. (No screen swop or single key BASIC).

Registered by Australia Post
Publication No. VBG4212

If undeliverable return to
KAOS, 10 Forbes St
Essendon, Victoria 3040

BTB